Practical validation of several fault attacks against the Miller algorithm

Ronan Lashermes, Marie Paindavoine, Nadia El Mrabet, Jacques Jean-Alain Fournier, Louis Goubin

Practical validation of several fault attacks against the Miller algorithm

Ronan Lashermes*†, Marie Paindavoine‡§, Nadia El Mrabet¶, Jacques J.A. Fournier*, Louis Goubin†

*CEA Tech, DPACA/LSAS, Gardanne, France.
firstname.lastname@cea.fr

†UVSQ-PRiSM, Versailles, France
firstname.lastname@prism.uvsq.fr

‡Orange Labs, Caen, France
firstname.lastname@orange.com

§U. Lyon CNRS ENSL INRIA UCBL LIP, Lyon, France
marie.paindavoine@ens-lyon.fr

¶LIASD - Université Paris 8, France
elmrabet@ai.univ-paris8.fr

***Abstract*—Pairing based cryptography (PBC) is touted as an efficient approach to address usability and privacy issues in the cyberspace. Like most cryptographic algorithms, PBC must be robust not only against theoretical cryptanalysis but also against practical physical attacks such as fault injections. The computation of the Tate pairing can be divided into two parts, the Miller Algorithm and the Final Exponentiation. In this paper, we describe practical implementations of fault attacks against the Miller Algorithm validating common fault models used against pairings. In the light of the implemented fault attacks, we show that some *blinding techniques* proposed to protect the algorithm against Side-Channels Analyses cannot be used as countermeasures against the implemented fault attacks.**

***Index Terms*—Pairing, Miller algorithm, fault attacks, blinding countermeasures, EM fault injection**

I. Introduction

Pairing based cryptography (PBC) is touted as an efficient approach to address usability and privacy issues in the cyberspace. As most cryptographic algorithms, PBC must be robust not only against theoretical cryptanalysis but also against practical physical attacks like fault injections. PBC uses bilinear mappings (or pairings) to construct cryptographic schemes for applications such as Identity-Based Encryption (IBE) [1], anonymous IBE, one round tripartite Diffie-Hellman key exchange or searchable encryption [2]. A pairing calculation consists of two major steps namely the *Miller Algorithm* (MA) followed by the *Final Exponentiation* (FE). An exhaustive literature is available on the choice of curves and associated parameters for secure and efficient PBC implementation [3]. Recent advances in solving the Discrete Logarithm Problem (DLP) in small characteristics [4] have reinforced the importance of focusing mainly on fields with a big prime characteristic. Issues linked to the resistance of PBC implementations against side channels and fault attacks have also been covered in the literature [5], [6]. Most of the proposed fault attack schemes against PBC are theoretical and focus on the Miller algorithm [7], [8], [9] for which associated countermeasures have been proposed.

In this paper we explain how fault attacks have been implemented in practice against the Miller Algorithm independently from the final exponentiation. In the light of the implemented attacks, we analyse the countermeasures proposed in the literature to show that using blinding methods to add randomness to the intermediate calculations are not always efficient to thwart our attack.

The paper is organized as follows. We first provide a quick background on pairings and an overview of the current state-of-art of fault attacks on the Miller algorithm. We describe the principle behind and the practical settings for the fault attacks that we implemented. We then show that some blinding countermeasures are

Marie Paindavoine's research for this paper was done during her internship with the CEA Tech.

Nadia El Mrabet wishes to acknowledge support from French project ANR INS 2012 SIMPATIC.

inefficient against our fault model and consequently look at how to properly secure the Miller algorithm.

II. Background on pairings

A. The Tate and Ate pairings

A pairing e is a bilinear and non degenerate map such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, with $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ three cyclic groups of the same prime order r. Let p be a prime number, E be an elliptic curve over $\mathbb{F}_p$ and r a prime divisor of card$(E(\mathbb{F}_p))$. Efficient algorithms for pairings are realized with $\mathbb{G}_1$, $\mathbb{G}_2$ subgroups of an elliptic curve $E(\mathbb{F}_{p^k})$ with point at infinity P_∞ and $\mathbb{G}_3$ is the subgroup of the r^{th} roots of unity in $\mathbb{F}_{p^k}$, where k is the smallest integer such that r divides $(p^k - 1)$. The integer k is called the embedding degree of E with respect to r.

The first pairings used in cryptography were the Weil and the Tate pairings [10], [11] and they have become a very active field of research with uses in innovative protocols like IBE [12]. The latest and more efficient implementations are the Ate [13], twisted pairings [13], improved with the notion of optimal pairings [14] and pairing lattices [15]. These pairings (that we call "Tate-like" pairings) are based on the model of the reduced Tate pairing defined by

$$\begin{cases} e_T : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) & \to & \mu_r \subset \mathbb{F}_{p^k}^* \\ (P, Q) & \to & f_{r,P}(Q)^{\frac{p^k-1}{r}} \end{cases},$$

where $f_{r,P}(Q)$ represents the Miller function computed using the Miller algorithm and defined by the divisor $r(P) - ([r]P) - (r-1)(P_\infty)$ evaluated at the point Q.

The Ate pairing has been developed in order to accelerate the computation time of a pairing, this "Tate-like" pairing is the one implemented and used in our experiments. Let π_p be the Frobenius endomorphism on a point given by $\pi_p(X, Y) = (X^p, Y^p)$. Let t be the trace of Frobenius ($\#E(\mathbb{F}_p) = p + 1 - t$). The Ate pairing is defined by $P \in E(\mathbb{F}_p)[r] \cap \ker(\pi_p - [1])$, $Q \in E(\mathbb{F}_{p^k})[r] \cap \ker(\pi_p - [p])$, $K = t - 1$ and $ate(Q, P) = f_{K,Q}(P)^{\frac{p^k-1}{r}}$.

Since the Miller algorithm is the central step for the most popular pairings, several optimizations have been proposed [16]. Algorithm 1 shows the Miller algorithm using the denominator elimination optimization. In PBC, if a secret is used, then it is one of the inputs of the pairing, either the point P or the point Q.
From Algorithm 1, f_1 shall be called the *Miller variable*, the *Doubling step* shall consist of the operations at lines 4 and 5 while the *Addition step* is represented by lines 8 and 9.

Algorithm 1: Miller algorithm for the Ate pairing

Data: $K = (K_n \dots K_0)_2$, $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$;
Result: $f_{K,Q}(P) \in \mathbb{G}_3$;

1 $T \leftarrow Q$;
2 $f_1 \leftarrow 1$;
3 **for** $i = n - 1$ **to** 0 **do**
4 $\quad T \leftarrow [2]T$;
5 $\quad f_1 \longleftarrow {f_1}^2 \times h_1(P)$;
6 $\quad h_1(x)$ is the equation of the tangent at the point T;
7 $\quad$ **if** $K_i = 1$ **then**
8 $\quad\quad T \leftarrow T + Q$;
9 $\quad\quad f_1 \longleftarrow f_1 \times h_2(P)$;
10 $\quad\quad h_2(x)$ is the equation of the line (QT);
11 $\quad$ **end**
12 **end**
13 **return** f_1

B. Barreto-Naehrig curves

Given the diversity of pairing implementations, it is not possible to study fault attacks on all of them. The 128-bit security level was chosen as usually recommended. The best current implementations with this security use Barreto-Naehrig (BN) curves [17]. Furthermore, the fault attack we describe here is independent from the elliptic curve parameters and from the coordinates used. As an example, we used the parameters specified in [18] to construct our curve and our fields.

A BN curve is defined by the parametrized values $t(x)$, $r(x)$ and $p(x)$ such that x fully defines the curve, with $r(x)$ and $p(x)$ prime integers. In our case $x = \texttt{0x3FC0100000000000}$ and $\mathbb{F}_{p^{12}}$ is constructed through the following tower extension:

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[u]/(u^2 - \beta), \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[v]/(v^3 - u), \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[w]/(w^2 - v), \end{aligned}$$

with $\beta = -5$ is a quadratic non-residue in $\mathbb{F}_p$, u is a cubic non-residue in $\mathbb{F}_{p^2}$, and v is a quadratic non-residue in $\mathbb{F}_{p^6}$.

The base for $\mathbb{F}_{p^{12}}$ as a vector space over $\mathbb{F}_{p^2}$ is then $(1, w, w^2, w^3, w^4, w^5)$ and the base for $\mathbb{F}_{p^{12}}$ over $\mathbb{F}_p$ is $(1, w, w^2, w^3, w^4, w^5, w^6, w^7, w^8, w^9, w^{10}, w^{11})$, and for $R \in \mathbb{F}_{p^{12}}$, we denote :

$$\begin{aligned} R = R_0 + R_1 w + R_2 w^2 + R_3 w^3 + R_4 w^4 + R_5 w^5 \\ + R_6 w^6 + R_7 w^7 + R_8 w^8 + R_9 w^9 + R_{10} w^{10} + R_{11} w^{11}, \end{aligned}$$

with $w^2 = v, w^6 = u$ and $w^{12} = \beta$.

The BN curve E that we use as an example is defined by the equation $Y^2 = X^3 + 5$ and the twisted curve E'

is given by $Y^2 = X^3 - 5/u = X^3 - u$.

III. Fault attacks against PBC

An overview of fault attacks against PBC is given in [19]. We briefly present here those attacks using the notations specified in Algorithm 1.

One of the first fault attacks on PBC was described by Page and Vercauteren [7] against the Duursma-Lee algorithm (efficient for supersingular elliptic curves over a finite field of characteristic 3 [20]). Page and Vercauteren use a random fault that occurs into the loop counter i of the Miller loop. The consequence is an observable difference in the number of iterations during the Miller algorithm. They recover the secret (point P or Q) by computing the quotient of two pairing results (one correct and one faulty). The quotient cancels out terms not influenced by the fault. Using the quotient and the equations of the Duursma-Lee algorithm, they solve a system thanks to the special form of the results of the Miller algorithm.

Whelan and Scott adopt a similar approach but with a different fault model [8]. They propose to inject a fault in the Miller variable f_1 itself. They add a random value to the function line evaluated during the last iteration of the Miller algorithm. The attack consists in analysing the quotient between a valid and a faulty result.

The attack of Page and Vercauteren was extended by El Mrabet to the more general case of the Miller algorithm for any elliptic curve [9]: the injected fault modifies the number of iterations of the Miller loop. Two faulty results corresponding to the Miller loop stopped at two consecutive iterations are needed. The author describes the recovery of the secret point using the quotient of those two intermediate results and the equations of the Miller algorithm.

More recently in [21], the authors proposed a fault model where the addition step is skipped with a fault attack. The instruction skip allows to recover h_2 which leads to an attack very similar to the previous ones mathematically.

To sum up, two approaches have been proposed to recover the value h_1: either by corrupting the data flow (data modification) or by modifying the control flow (loop skip, test skip). Once we have h_1, the Miller algorithm can be reversed to find the secret point.

Only one of the two input points is considered secret and the other one is public. This setting is notably representative of the level of knowledge which an attacker would have in an IBE scheme [1].

IV. Implemented fault attacks & results

Implementing a fault attack against the Miller algorithm consists of two steps: first finding h_1 which is in turn used to calculate the secret point. In this section, we illustrate how h_1 was found using faults before detailing how the secret point in the Miller algorithm can be recovered mathematically knowing h_1 (from [9]).

A. Experimental set-up

The attacker wants to find $h_1(P)$ at a known iteration. We use Electromagnetic (EM) pulses to generate faults in a microcontroller computing an Ate pairing. The platform used is similar to the one detailed in [22], [23].

The target of our experiment is a Cortex-M3 32−bit microcontroller running at 56 MHz on which we run our own implementation of an Ate pairing following the algorithms in [18] and coded in C language. The chip has not been designed as a secure chip but it bears some basic sensors for monitoring power and clock glitches which trigger hardware interrupts. These sensors are active during the experiments and are able to detect the EM pulses if they are too powerful. The board is underpowered at 2.8 V instead of 3.3 V in order to increase the sensitivity of the chip to the EM pulses.

The EM pulse injection platform used is composed of a pulse generator, a handmade coil antenna of diameter 1 mm (seen on Fig. 1), an XYZ motorized stage allowing to precisely ($\pm 1\,\mu$m) position the targeted chip with respect to the antenna, all this within a Faraday cage and with a computer controlling all those devices.

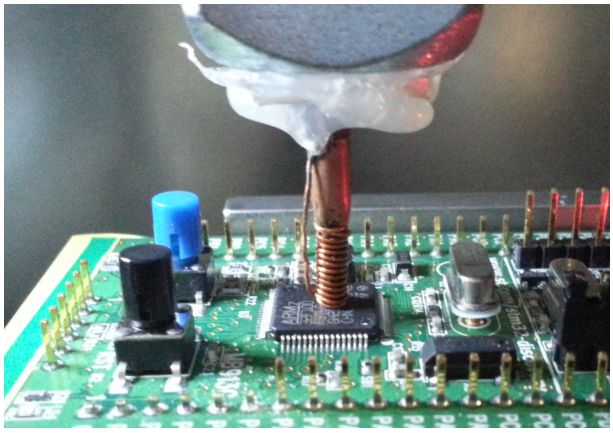

Fig. 1. EM bench: the chip and the probe.

At a precise moment determined by the attacker, the generator sends a voltage pulse, between −210 V and 210 V, to the coil antenna with rising and falling times of 2 ns and a duration of 10 ns. The coil is precisely

placed just above the targeted position on the circuit. The EM pulse reaches the Power Ground Network of the chip and alters the computation performed by the device.

In order to find the right parameters of the fault injection set-up, we control and communicate with the chip through Keil µvision [24] and its UVSOCK library. This allows us to access various internal registers giving us insights of what is going on after a fault injection even if an interrupt has been raised (i.e. the glitch has been detected). Experiments consist in finding the right parameters: position of the coil with respect to the chip, pulse amplitude, and the instant of the pulse emission with respect to a trigger raised by the targeted chip when reaching the operations of interest.

First we have to determine the locations where faults can be injected without interrupting the execution flow of the code. To do so, we vary the XY position of the EM probe over the chip's surface and observe how the chip behaves under an EM pulse stress. Three characteristic behaviours have been observed, depending on the spatial XY location, as illustrated in Fig. 2: either no faults have been detected (the white strips in the figure); or interrupts are generated (the red/dark spots on the figure); or undetected faults have been inserted (the blue/light spots on the figure).

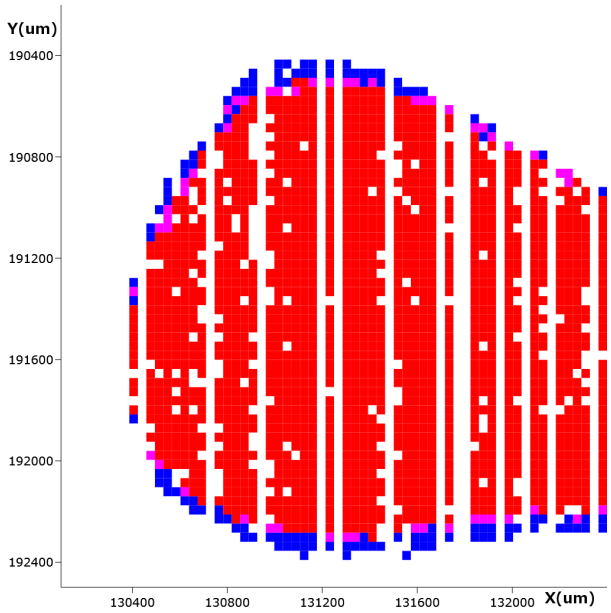

Fig. 2. XY map: red (dark) corresponds to interrupts and blue (light) to undetected faults (the white stripes and dots are visual artefacts where no faults have been injected).

With this approach we managed to find 2 set-ups for fault models corresponding to the modification of the control flow and that of the data flow (for recovering h_1):

- *Controlled add* model: a controlled value is added to h_1 through a fault injection. "Controlled" means that the value should be manageable: not totally random but with only a limited uncertainty allowing to individually test all possibilities. This fault model typically allowed us to implement the fault attack described in [6].
- *Loop skip* model: the last iteration of the Miller algorithm is skipped. This fault model suits the needs for the fault attacks described in [7] and [9].

Both these models have been theoretically proposed for fault attacks on pairings and allow to get h_1. Since our implementation is an Ate pairing, the last iteration of the Miller algorithm has only a doubling step and no addition step.

B. First fault attack model: controlled add

In this attack, the secret point is Q and P is known to the attacker. In this experiment, we inject a fault during an addition in $\mathbb{F}_p$. The latter operation requires a multi-word addition algorithm on the 32-bit chip.

We target one word of the addition and vary the instant of the injection of the pulse.

During a campaign (with carefully chosen parameters, seen on Fig. 3) of 1014 pulses emitted by the EM bench, 373 (37%) pulses induced an interrupt and 172 (17%) undetected faults were created on the least significant word. No fault was created on the other words, apart from the carry propagation due to the faulted least significant word.

The values obtained are:

- Correct value: $0x6EBCDA28$
- $0x6EBCDA29$
- $0xB75E6E10 \approx 0x6EBCDA28/2$
- $0xF75F792B$
- $0xFFFFFFFF$

It is difficult to exactly explain what the exact effect of the electromagnetic pulses is since we do not have inside knowledge of the inner workings of the chip. We guess that in this case, the pulse affects the bus when data are retrieved from the RAM. The synchronisation may be altered which leads to 'shifts' within a word, or to the use of the bus pre-charge value of $0xFFFFFFFF$.

It is possible to use a fault on the modular addition to recover $h_1(P)$ if the attacker knows the point P. In this case, he can target the evaluation of one of the coordinates (R_0, R_3 or R_4) of $h_1(P)$ during the last iteration of the Miller algorithm. For example the

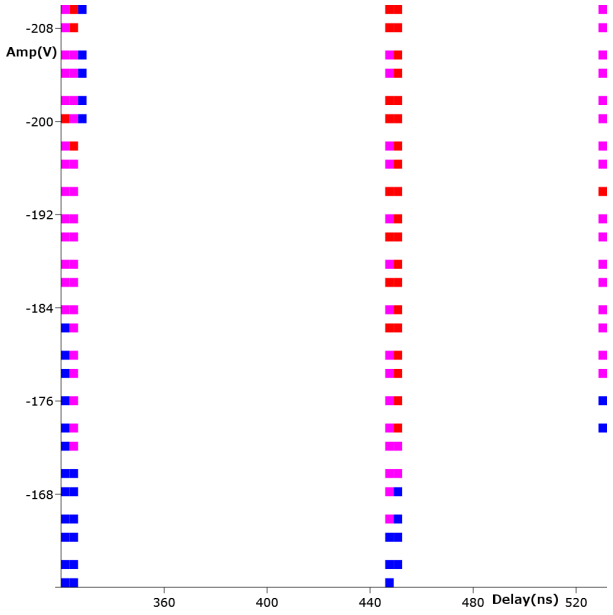

Fig. 3. Delay (horizontal) and stress (vertical) variations: red (dark) corresponds to interrupts and blue (light) to undetected faults.

value R_0 is computed with an algorithm ending with this pseudo-C code ($t_0 \in \mathbb{F}_{p^2}$):

```
t0 = t0 + t0; //fast modular doubling
R0 = t0 * YP; //P = (XP : YP)
```

The attacker can now recover $h_1(P)$ by injecting a known fault e on the modular addition giving $t_0^* = t_0 + e$. This fault is propagated onto R_0:

$$R_0^* = t_0^* \cdot YP = (t_0 + e) \cdot YP = R_0 + e \cdot YP = R_0 + \Delta_{R_0}. \quad (1)$$

Since e and YP are known to the attacker, Δ_{R_0} is known as well.
At the last iteration of the Miller algorithm, we have:

$$f_{K,Q}(P) = f_1^2 \times h_1(P) \quad (2)$$

If the attacker is able to inject a known fault Δ_{R_0} in $h_1(P)$, he recovers

$$f_{K,Q}(P)^* = f_1^2 \times (h_1(P) + \Delta_{R_0}). \quad (3)$$

As he knows $f_{K,Q}(P)^*$, $f_{K,Q}(P)$ and Δ_{R_0}, he can find $h_1(P)$:

$$h_1(P) = \frac{f_{K,Q}(P) \times \Delta_{R_0}}{f_{K,Q}(P)^* - f_{K,Q}(P)} \quad (4)$$

If P is the secret and Q is known it is possible to obtain the same result with a fault on the last operation computing R_3 which is a modular subtraction.

C. Second fault attack model: loop skip

The goal of this experiment is to skip one, or several, iterations of the Miller algorithm. The operations of interest are the ones which update the loop counter and decide whether to quit the loop or not.

Fig. 4. Assembly code for the loop test

```
SUBS    r4,r4,#1    //index decrement
CMP     r4,#0x00    //loop test: is r4 == 0
BGE     0x080008C2  //>= branch to start
```

NOP instructions have been added into the code when the trigger is raised in order to have an easier synchronisation between our code and the EM pulse (there is an incompressible delay of 80 ns or about 5 instructions). Additionally, the NOP instructions also clear the pipeline which lead to a better reproducibility of the faults. Our tests without the NOP instructions made the synchronisation work a lot harder and when a fault is injected, the probability of repeating it (with all parameters fixed) is lower. In order to have a clear presentation, the NOP instructions have been kept through the experiments in this paper. Yet, it is possible to remove them at the cost of harder experiments (and/or better equipment).

When injecting EM pulses during the execution of operations on the loop counter, several behaviours have been observed:

- Hardware crash: the chip is not responding and often needs to be reprogrammed.
- Software crash: the chip is in an incoherent state (e.g. after a branch to a random location).
- Interrupt: a problem has been detected by the chip which stops the normal course of the program.
- Computation error: the program has performed a mistake but the chip is unaware of it.
- Normal execution: the program has been normally executed.

We fine-tuned our settings to perform *loop skips* in a "computation error" scenario: to do so, we first used a complete Miller algorithm in which we have included several counters which act as "snow-steps", the value of the counter easily telling us whether that instruction has been executed or not. We used these counters to identify the correct time of injection while maintaining the pulse amplitude at -210 V thus creating the highest stress. At the right time of injection, we could only observe interrupts at first. Finally by varying slightly the time of injection and the stress similarly to what can be seen on Fig. 3, we were able to inject a fault which induces the "skipping" of the branch instruction.

In the Ate pairing, the last iteration is a tangent evaluation only:

$$f_{K,Q}(P) = f_1^2 \times h_1(P). \quad (5)$$

Thus if we skip the last iteration, we obtain

$$f_{K,Q}(P)^* = f_1. \quad (6)$$

Finally, h_1 is simply

$$h_1(P) = \frac{f_{K,Q}(P)}{(f_{K,Q}(P)^*)^2}. \quad (7)$$

By simply changing the targeted iteration, we are in fact able to leave the loop at whatever iteration we choose by skipping the corresponding branch instruction. We confirmed this by skipping the last two iterations simply by raising the trigger exactly one iteration earlier. If it is possible to exit the loop at whatever iteration the attacker desires, the best choice would be to execute only the first iteration of the Miller algorithm. In this case

$$f_{K,Q}(P)^* = h_1(P), \quad (8)$$

it becomes unnecessary to compute a correct result of the Miller algorithm. This trick may be useful since accessing to the correct value of the Miller algorithm is in practice a feat, due to the final exponentiation.

D. Recovering the secret point from h_1

Now that we have seen how the value of $h_1(P)$ was recovered in practice, we are going to illustrate how the secret point was derived from the latter value. We know that $Q \in E(\mathbb{F}_{p^k})$ and $P \in E(\mathbb{F}_p)$. In practice however we use the degree 6 twisted curve for the representation of Q: we simplify the notation by denoting Q as the point $(x_q : y_q) \in E'(\mathbb{F}_{p^2})$ instead of $(x_q \cdot w^2 : y_q \cdot w^3) \in E(\mathbb{F}_{p^k})$. Additionally, the point Q (and therefore T) is represented in jacobian coordinates $(X_Q : Y_Q : Z_Q)$ which map to the affine representation $(X_Q/Z_Q^2 : Y_Q/Z_Q^3)$. The attacker can recover $h_1(P)$ for the Ate pairing with

$$h_1(P) = (3X_T^3 - 2Y_T^2) \cdot w^6 + 2Y_T Z_T^3 y_p \cdot w^3 - 3X_T^2 Z_T^2 x_p \cdot w^4, \quad (9)$$

$$h_1(P) = R_0 + R_3 \cdot w^3 + R_4 \cdot w^4. \quad (10)$$

with $R_0, R_3, R_4 \in \mathbb{F}_{p^2}$ (since $w^6 = u \in \mathbb{F}_{p^2}$) recovered through identification and $T = [i]Q$ for some i known to the attacker.

For the Ate pairing R_0, R_3, R_4 provide a system in $\mathbb{F}_{p^2}$:

$$R_0 = (3X_T^3 - 2Y_T^2) \cdot u, \quad R_3 = 2Y_T Z_T^3 y_p,$$
$$R_4 = -3X_T^2 Z_T^2 x_p.$$

First, if P (for the Ate pairing) is the secret and Q is known, P can trivially be obtained with this system since $T = [i]Q$ is known to the attacker and the system is linear in P coordinates. That is why from now on we focus on the case where the secret point is Q while P is known. The solution is barely more complex. The system now yields the univariate polynomial

$$\frac{R_0^2}{\beta} \cdot Z_T^{12} + \left(4\frac{R_0}{u}\lambda_2^2 - 9\lambda_3^3\right) \cdot Z_T^6 + 4\lambda_2^4 = 0 \quad (11)$$

with $\lambda_2 = \frac{R_3}{2y_p}$ and $\lambda_3 = -\frac{R_4}{3x_p}$. This polynomial can be solved on $\mathbb{F}_{p^2}$ providing the value of Z_T.

Once we know Z_T, we use it into the initial system to obtain X_T and Y_T. The points which do not lie on the curve are eliminated. Finally, the possibilities for $Q = [i^{-1}]T$ are computed.

If it is possible to control the loop at which the Miller algorithm returns its result, one would prefer the result after the first iteration. In this case no correct execution is needed to extract h_1 since $f_{0,Q}(P) = 1$. This case was practically achieved just by moving the trigger to the first iteration.

E. The Final Exponentiation

The attacker must be able to access the faulty Miller results in our fault attack, which can be difficult in practice because of the presence of the final exponentiation for BN curves.

Several strategies have been proposed to deal with this final exponentiation. A scan chain attack has been proposed in [9] to directly access the result of the Miller Algorithm. More recently [25], a scheme has been proposed whereby fault attacks can be used to reverse the final exponentiation. By showing that the Miller algorithm is vulnerable in practice, we open the way to a practical fault attack on the whole pairing with double faults. Yet how to properly override the Final Exponentiation in conjunction with a fault attack on the Miller Algorithm remains an open problem which has to be further studied, but this is not within the scope of this paper.

V. Analysis of the countermeasures

Once we have tested and demonstrated the practical feasibility of fault attacks on the MA, we hereby analyse some blinding countermeasures proposed in the literature and find out the *loop skip* fault model forbids to use the blinding countermeasures proposed against side-channel analyses as countermeasures against fault attacks.

A. Coordinates randomization

As proposed in [26] against side-channels analyses, instead of executing the Miller Loop with $Q = (X_Q : Y_Q : Z_Q) \in E(\mathbb{F}_{p^2})$ (represented in jacobian coordinates), we execute it with

$$Q = (\lambda^2 X_Q : \lambda^3 Y_Q : \lambda Z_Q),$$

where $\lambda \in \mathbb{F}_{p^2}$ is a random blinding value. We note $h_1^{(\lambda)}, h_2^{(\lambda)}, f_{K,P}^{(\lambda)}$ the line evaluations and the output of the Miller algorithm when performed with this blinding. When we include the λ in the tangent equation, we find that it is possible to factor it. For the doubling step the equation becomes $h_1^{(\lambda)} = \lambda^{24i} h_1$, and for the addition step it becomes $h_2^{(\lambda)} = \lambda^{9i+12} h_2$, where i is an integer related to the number of iterations. The value of i can be found, but since this value has no influence on the result of the attack we remove it. Thus the result of the Miller loop is $f_{K,Q}^{(\lambda)} = \lambda^a \cdot f_{K,Q}$ for some integer a.

In order to perform the fault attack, we need two different executions of the Miller algorithm. Thus we use two different blinding values, one for each execution. But actually, this adds only one unknown into the system:

$$\begin{aligned} h_1(P)^{(\lambda)} &= \frac{f_{K,Q}^{(\lambda_1)(\tau+1)}(P)}{f_{K,Q}^{(\lambda_2)(\tau)}(P)^2} \\ &= \frac{\lambda_1^a \cdot f_{K,Q}^{(\tau+1)}(P)}{\lambda_2^b \cdot f_{K,Q}^{(\tau)}(P)^2} = \frac{\lambda_1^a}{\lambda_2^b} \cdot h_1(P). \quad (12) \end{aligned}$$

We name $L = \frac{\lambda_1^a}{\lambda_2^b}$ the new unknown and hence have:

$$\begin{aligned} h_1(P)^{(\lambda)} = L \cdot ((3X_T^3 - 2Y_T^2) \cdot w^6 + 2Y_T Z_T^3 y_p \cdot w^3 \\ -3X_T^2 Z_T^2 x_p \cdot w^4), \quad (13) \end{aligned}$$

$$h_1(P)^{(\lambda)} = R_0^{(\lambda)} + R_3^{(\lambda)} w^3 + R_4^{(\lambda)} w^4. \quad (14)$$

By identification of the decomposition in $\mathbb{F}_{p^2}$, we obtain the system

$$R_0^{(\lambda)} = L \cdot R_0, R_3^{(\lambda)} = L \cdot R_3, R_4^{(\lambda)} = L \cdot R_4. \quad (15)$$

To this system, we add the equation derived from the fact that T lies on the curve E:

$$Y_T^2 = X_T^3 + 5Z_T^6. \quad (16)$$

The system (15) and the equation (16) can be solved for the Ate pairing. To be solved, the resulting system requires the computation of the Gröbner basis which provides an equation of degree 12 in Z_T only.

B. Miller's variable blinding

In this countermeasure again side-channels analyses, the line evaluation value is multiplied by a random element L of $\mathbb{F}_{p^2}$ for all iterations as suggested by Scott in [27]. Thus, we have:

$$h_1^{(\lambda)} = L \cdot h_1. \quad (17)$$

As can be seen immediately, this countermeasure can be bypassed in the exact same way as the previous one with the system (15) and equation (16).

C. Protecting the Miller algorithm

In order to protect efficiently the Miller algorithm, countermeasures should take into account the fault models. We have shown that the *loop skip* model allows to bypass the *coordinates randomization* and *Miller's variable blinding* countermeasures even if the attacker is not able to replay a mask already used. Two other blinding techniques have been proposed in the literature, namely the additive blinding and the multiplicative blinding which are effective if the mask values are properly changed at each execution.

In the additive blinding technique [7], a mask is added on one of the input points and a second pairing is computed to remove the mask:

$$e(Q, P) = e(Q, P + M)e(Q, M)^{-1}$$

if Q is the secret point.

Another technique, called the multiplicative blinding [7], moves the security concern away from the pairing computation. Let a be a random value modulo r and b such that $a \cdot b = 1 \mod r$. Then $e(Q, P) = e([a]Q, P)^b$. It appears that even if the attacker is able to recover $[a]Q$, he cannot find the secret point Q thanks to the ECDLP. Now the security of the system relies on the secrecy of a and b.

The protection of the Miller algorithm against the *loop skip* fault attack can also be realized with the protection of the loop itself. It can be enough to securely ensure that the correct number of iterations have been computed, i.e. by verifying the counter or with timing monitoring for examples. Moreover adding 'classical' countermeasures causing de-synchronisations (random executions within the Miller Algorithm) would complicate the experimental set-up of the attacks proposed in this paper.

VI. Conclusion

In this paper we describe how two (theoretical) fault attack models proposed in the literature against the Miller algorithm of a pairing calculation have been validated in practice against an Ate pairing implemented

on a 32-bit general purpose processor. To our best knowledge, this is the first paper that validates in practice the fault attacks described against the MA. Such attacks do not put at risk the entire pairing calculation so long as the final exponentiation is considered "fault-proof" but it nevertheless proves that most of the theoretical attack schemes described in the literature against the MA can be implemented. In the light of the results obtained, we review the blinding countermeasures proposed in the literature and show that some of them are inefficient. This work shows that those blinding countermeasures should not be used on their own to protect the Miller Loop against fault attacks. Future work shall consist in testing other attack routes described against the Miller algorithm and other countermeasures. A complete fault attack on a pairing taking into account the final exponentiation would be of interest both theoretically and experimentally.

REFERENCES

[1] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil pairing," SIAM J. of Computing, vol. 32, no. 3, pp. 586–615, 2003.

[2] R. Dutta, R. Barua, and P. Sarkar, "Pairing-Based Cryptographic Protocols : A Survey," Cryptology ePrint Archive, Report 2004/064, 2004.

[3] D. Freeman, M. Scott, and E. Teske, "A taxonomy of pairing-friendly elliptic curves," J. Cryptology, vol. 23, no. 2, pp. 224–280, 2010.

[4] A. Joux, "A new index calculus algorithm with complexity l(1/4+o(1)) in very small characteristic," Cryptology ePrint Archive, Report 2013/095, 2013.

[5] N. El Mrabet, G. Di Natale, M.-L. Flottes, B. Rouzeyre, and J.-C. Bajard, "Differential Power Analysis against the Miller Algorithm," Tech. Rep., Aug. 2008, prime 2009, IEEE Xplore.

[6] C. Whelan and M. Scott, "Side channel analysis of practical pairing implementations: Which path is more secure?" in VIETCRYPT 2006, ser. LNCS. Springer, 2006, vol. 4341, pp. 99–114.

[7] D. Page and F. Vercauteren, "A Fault Attack on Pairing-Based Cryptography," Computers, IEEE Transactions on, vol. 55, no. 9, pp. 1075 –1080, sept. 2006.

[8] C. Whelan and M. Scott, "The Importance of the Final Exponentiation in Pairings when considering Fault Attacks," in Pairing 2007, ser. LNCS. Springer, 2007, vol. 4575, pp. 225–246.

[9] N. El Mrabet, "What about Vulnerability to a Fault Attack of the Miller's algorithm During an Identity Based Protocol?" in Advances in Information Security and Assurance, ser. LNCS. Springer, 2009, vol. 5576, pp. 122–134.

[10] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," in STOC '91, ACM, Ed., 1991, pp. 80–89.

[11] G. Frey and H. Ruck, "A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of groups," in Mathematics of Computation, vol. 62. American Mathematical Society, 1994, pp. 865–874.

[12] M. Joye and G. Neven, Eds., Identity Based Cryptography. IOS Press, 2008.

[13] F. Hess, N. P. Smart, and F. Vercauteren, "The Eta Pairing Revisited," in IEEE Transactions on Information Theory, vol. 52, 2006, pp. 4595–4602.

[14] F. Vercauteren, "Optimal pairings," IEEE Trans. Inf. Theor., vol. 56, no. 1, pp. 455–461, Jan. 2010.

[15] F. Hess, "Pairing lattices," in Pairing, ser. LNCS, S. D. Galbraith and K. G. Paterson, Eds., vol. 5209. Springer, 2008, pp. 18–38.

[16] M. Scott, "On the efficient implementation of pairing-based protocols," in IMA International Conference, ser. LNCS, L. Chen, Ed., vol. 7089. Springer, 2011, pp. 296–308.

[17] P. Barreto and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order," in Selected Areas in Cryptography, ser. LNCS. Springer, 2006, vol. 3897, pp. 319–331.

[18] J.-L. Beuchat, J. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya, "High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves," in Pairing 2010, ser. LNCS. Springer, 2010, vol. 6487, pp. 21–39.

[19] N. El Mrabet, D. Page, and F. Vercauteren, "Fault attacks on pairing-based cryptography," in Fault Analysis in Cryptography, ser. Information Security and Cryptography. Springer, 2012, pp. 221–236.

[20] I. M. Duursma and H.-S. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," in ASIACRYPT, ser. LNCS, C.-S. Laih, Ed., vol. 2894. Springer, 2003, pp. 111–123.

[21] K. Bae, S. Moon, and J. Ha, "Instruction fault attack on the miller algorithm in a pairing-based cryptosystem," in IMIS, 2013, pp. 167–174.

[22] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz, "Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller," in Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on, Aug 2013, pp. 77–88.

[23] A. Dehbaoui, J.-M. Dutertre, B. Robisson, and A. Tria, "Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES," in FDTC. IEEE, 2012, pp. 7–15.

[24] ARM, Keil uvision, `http://www.keil.com/uvision/`.

[25] R. Lashermes, J. Fournier, and L. Goubin, "Inverting the final exponentiation of tate pairings on ordinary elliptic curves using faults," in CHES 2013, ser. LNCS. Springer, 2013, vol. 8086, pp. 365–382.

[26] T. Kim, T. Takagi, D.-G. Han, H. Kim, and J. Lim, "Side channel attacks and countermeasures on pairing based cryptosystems over binary fields," in Cryptology and Network Security, ser. LNCS. Springer Berlin Heidelberg, 2006, vol. 4301, pp. 168–181.

[27] M. Scott, "Computing the tate pairing," in CT-RSA 2005, ser. LNCS, A. Menezes, Ed. Springer, 2005, vol. 3376, pp. 293–304.